EL-213

ANALOG CIRCUITS

OPEN DOOR FROM REMOTE LOCATION

GROUP : 5

Assigned By : Prof. Rutu Parekh

GROUP MEMBERS:

NAME	STUDENT ID
ABHISHEK SUTARIYA	201701012
JEEL PATEL	201701176
ABHAY MISTRY	201701178
PRIYANK SATASIYA	201701183
NIRAJ KAMDAR	201701184
VINAY PARMAR	201701205
PARTH CHAUHAN	201701210
KASHYAPKUMAR DAXINI	201701411
JAY MEHTA	201701446
ANIRUDHDH BERIYA	201701461

Contents

Acknowledgement

We would like to express our special thanks of gratitude to our respected **Prof. Rutu Parekh** who gave us the golden opportunity to do this wonderful project on the topic **Open door from remote location**, which also helped us in doing a lot of Research and we came to know about so many new things. We are really thankful to her.

Secondly we would like to thank our **Teaching Assistants and our batch mates** who constantly helped us throughout the project.

In the last but not the least we would like to express our thanks of gratitude to **everyone who directly or indirectly** helped us throughout the project.

INTRODUCTION

The internet shrunk the world by providing efficient interaction among devices located at remote locations which emerged as Internet of Things (IoT). In the era of Internet of Things (IoT) and Automation, we often want to control system remotely. In some cases we need to perform some changes in the system depending on some parameters. IoT devices are useful in security systems like door look, surveillance-camera, Home Automation, controlling industrial machines etc. All these IoT systems has basically three building blocks: i) sensors ii) microcontroller iii) actuators. Sensors collects data from surrounding, while microcontroller takes appropriate action according to event triggered by user or sensors, and actuators perform mechanical work.

In comparison to traditional door lock, smart door locks are more secure, aesthetically pleasing, and perfect for the elderly or disabled. Door lock involved in this project has two major hardware components: i) NodeMCU as microcontroller ii) Servomotor as actuator. NodeMCU has Wi-Fi connectivity, so one can control Servomotor via Internet with NodeMCU, which in turn controls the state of the door lock.

OBJECTIVE

Objective of the project is to achieve more security because smart door lock does not have a place for a key. This prevents break-ins because burglars are unable to pick or ‘bump’ the lock. Criminal methods of breaking and entering are improving and the majority of criminals can pick an ordinary key lock.

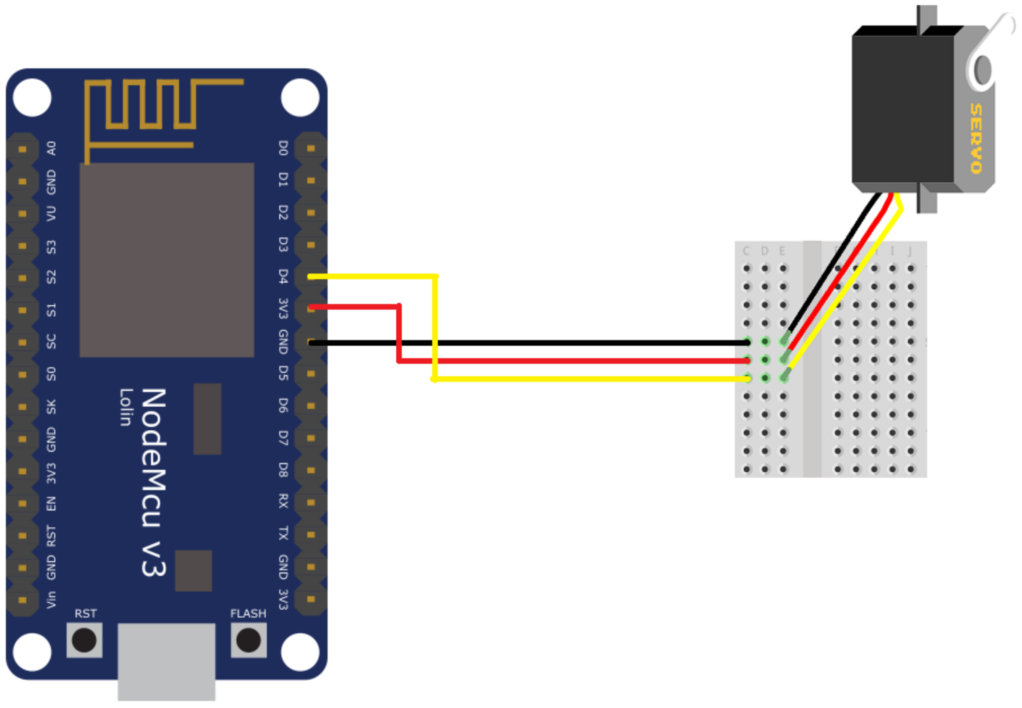
You won’t have to carry around a large set of keys and they will be less likely to be lost or stolen. Also, if you are a landlord, you don’t have to give residents keys or replace them if they lose them.

One may want someone else to enter the house when owner is not present, in such cases owner might not have to depend on neighbor or relative for the keys, just a contact would be enough to open the smart lock.

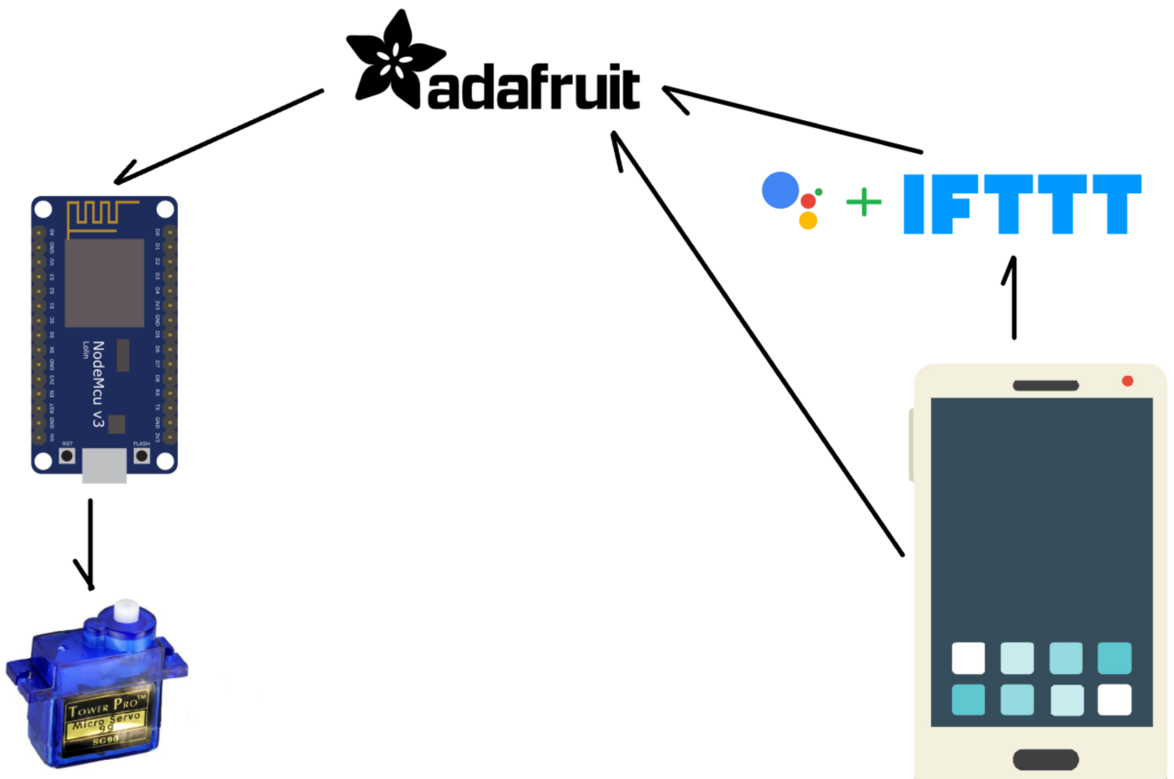
Technology has made living much easier for disabled or elderly people. And smart door locking system is no exception in that! Since it can be opened with smart phones, a senior citizen or disabled don’t have to struggle with keys or go near the door to open it.

Virtual keys give you a valuable security option to lock and unlock your doors easily, furthermore you can load various options such as turning on lights when door opens and etc. that key can be interacted via a free app on your Android or Apple device. A single smartphone application is able to control your locks and other smart home devices simultaneously.

CIRCUIT DIAGRAM



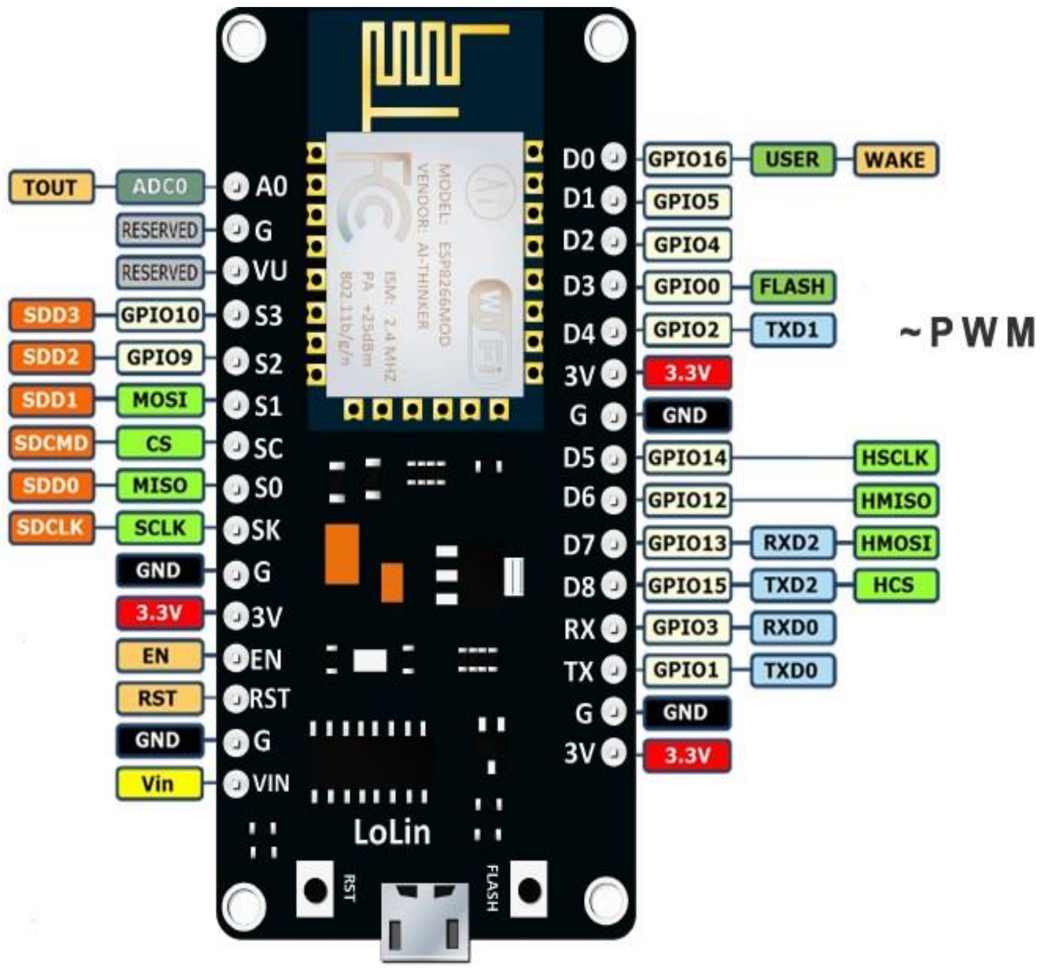
BLOCK DIAGRAM



COMPONENTS AND TOOLS

NodeMCU

NodeMCU is an open source LUA based firmware developed for ESP8266 Wi-Fi chip. By exploring functionality with ESP8266 chip, NodeMCU firmware comes with ESP8266 Development board/kit i.e. NodeMCU Development board.



Since NodeMCU is open source platform, their hardware design is open for edit/modify/build.

NodeMCU Dev Kit/board consist of ESP8266 Wi-Fi enabled chip. The **ESP8266** is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol.

NodeMCU Dev Kit has **Arduino like** Analog (i.e. A0) and Digital (D0-D8) pins on its board.

It supports serial communication protocols i.e. UART, SPI, I2C etc.

Using such serial protocols we can connect it with serial devices like I2C enabled LCD display, Magnetometer HMC5883, MPU-6050 Gyro meter + Accelerometer, RTC chips, GPS modules, touch screen displays, SD cards etc.

NodeMCU Dev Kit v1.0 pin descriptions

GPIO (General Purpose Input Output) Pins:

NodeMCU has general purpose input output pins on its board as shown in above pinout diagram. We can make it digital high/low and control things like LED or switch on it. Also, we can generate PWM signal on these GPIO pins.

ADC (Analog to Digital Converter) channel (A0):

NodeMCU has one ADC channel/pin on its board. Analog to Digital Converter (ADC) is used to convert analog signal into digital form. ESP8266 has inbuilt 10-bit ADC with only one ADC channel i.e. it has only one ADC input pin to read analog voltage from external device/sensor.

SPI (Serial Peripheral Interface) Pins:

NodeMCU based ESP8266 has Hardware SPI (HSPI) with four pins available for SPI communication. It also has SPI pins for Quad-SPI communication. With this SPI interface, we can connect any SPI enabled device with NodeMCU and make communication possible with it.

I2C (Inter-Integrated Circuit) Pins:

NodeMCU has I2C functionality support on ESP8266 GPIO pins. I2C (Inter Integrated Circuit) is serial bus interface connection protocol. It is also called as TWI (two wire interface) since it uses only two wires for communication. Those two wires are SDA (serial data) and SCL (serial clock).

UART (Universal Asynchronous Receiver Transmitter) Pins:

NodeMCU based ESP8266 has two UART interfaces, UART0 and UART1. Since UART0 (RXD0 & TXD0) is used to upload firmware/codes to board, we can't use them in applications while uploading firmware/codes.

How to write codes for NodeMCU?

After setting up ESP8266 with Node-MCU firmware, let's see the IDE (Integrated Development Environment) required for development of NodeMCU.

NodeMCU with ESPlorer IDE

Lua scripts are generally used to code the NodeMCU. Lua is an open source, lightweight, embeddable scripting language built on top of C programming language.

NodeMCU with Arduino IDE

Here is another way of developing NodeMCU with a well-known IDE i.e. Arduino IDE. We can also develop applications on NodeMCU using Arduino development environment. This makes easy for Arduino developers than learning new language and IDE for NodeMCU.

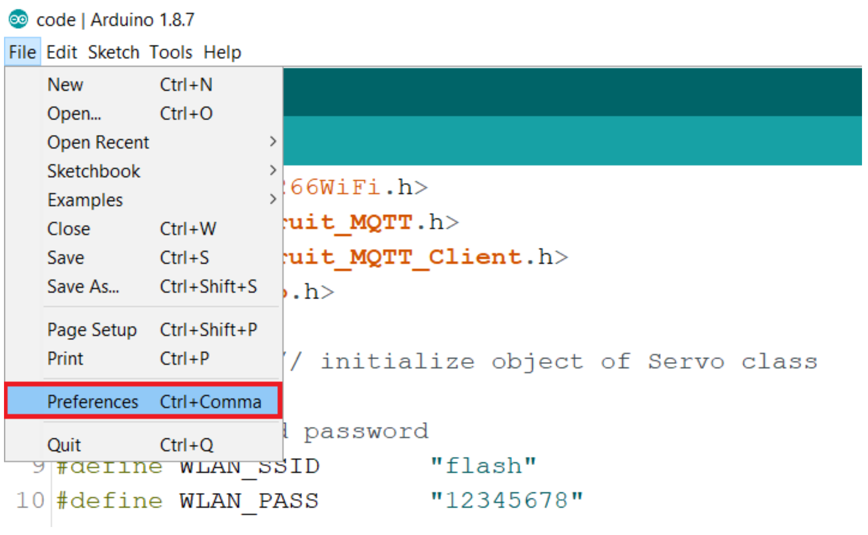
NodeMCU with Arduino IDE

NodeMCU is Lua based firmware of ESP8266. Generally, ESPlorer IDE is referred for writing Lua scripts for NodeMCU. It requires to get familiar with ESPlorer IDE and Lua scripting language.

Let's see about setting up Arduino IDE with NodeMCU.

First **Download Arduino IDE (version 1.6+)** https://www.arduino.cc/en/Main/Software

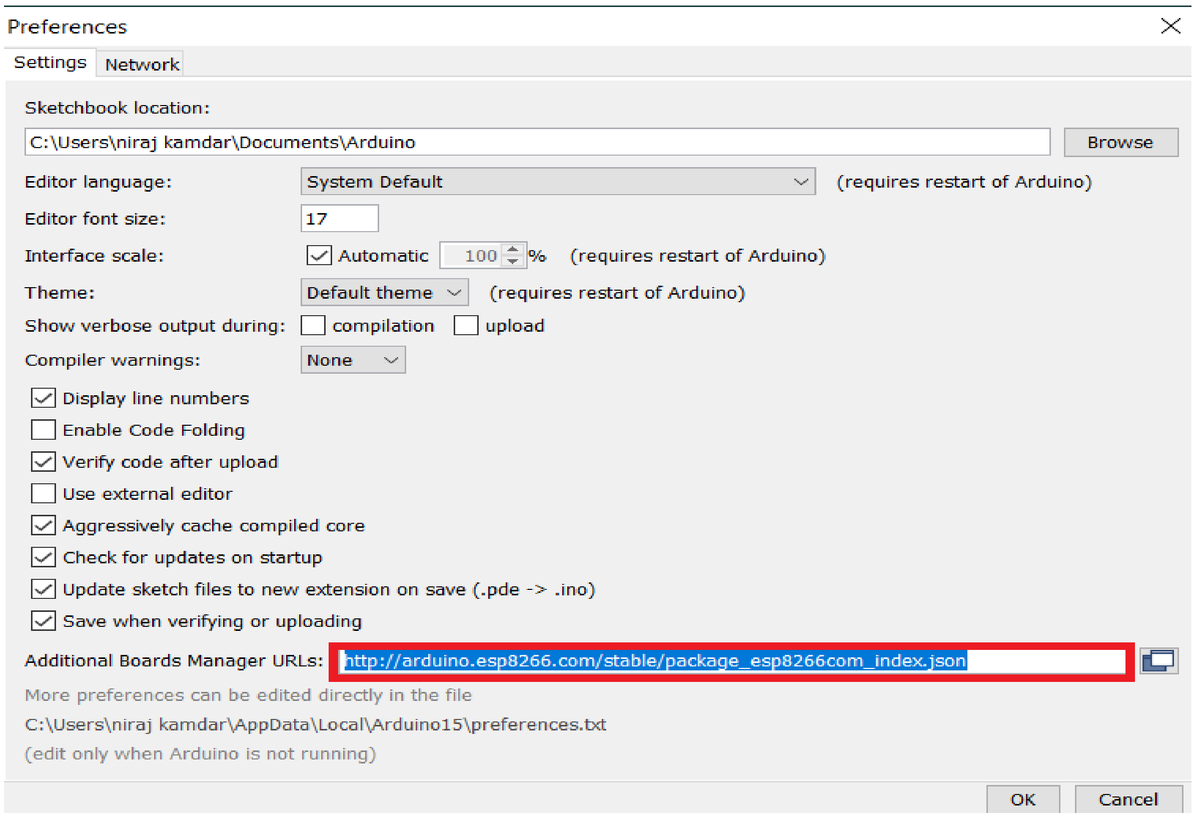
- **Open Arduino IDE** and **Go to File -> Preference**.



```
266WiFi.h>
uit_MQTT.h>
uit_MQTT_Client.h>
.h>
/ initialize object of Servo class
 password
#define WLAN_SSID       "flash"
#define WLAN_PASS       "12345678"
```

- Now on Preference window, **Enter below link in Additional Boards Manager URLs**

  http://arduino.esp8266.com/stable/package_esp8266com_index.json



Sketchbook location: C:\Users\niraj kamdar\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 17

Interface scale: [✓] Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: [ ] compilation [ ] upload

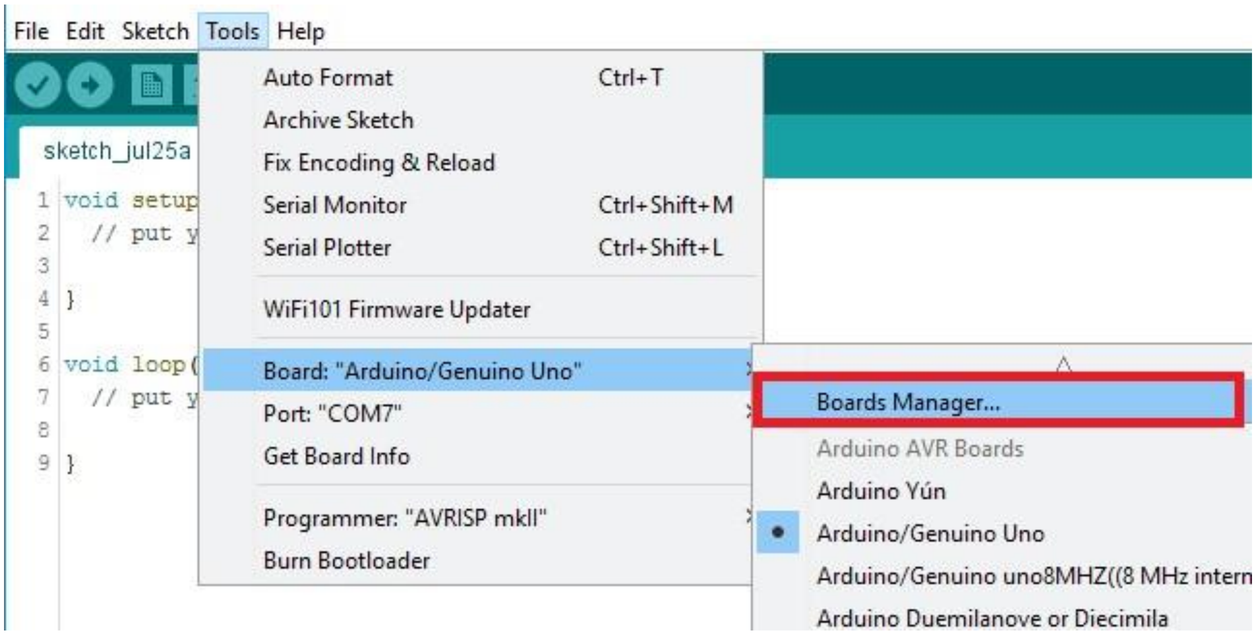
Compiler warnings: None

[✓] Display line numbers
[ ] Enable Code Folding
[✓] Verify code after upload
[ ] Use external editor
[✓] Aggressively cache compiled core
[✓] Check for updates on startup
[✓] Update sketch files to new extension on save (.pde -> .ino)
[✓] Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

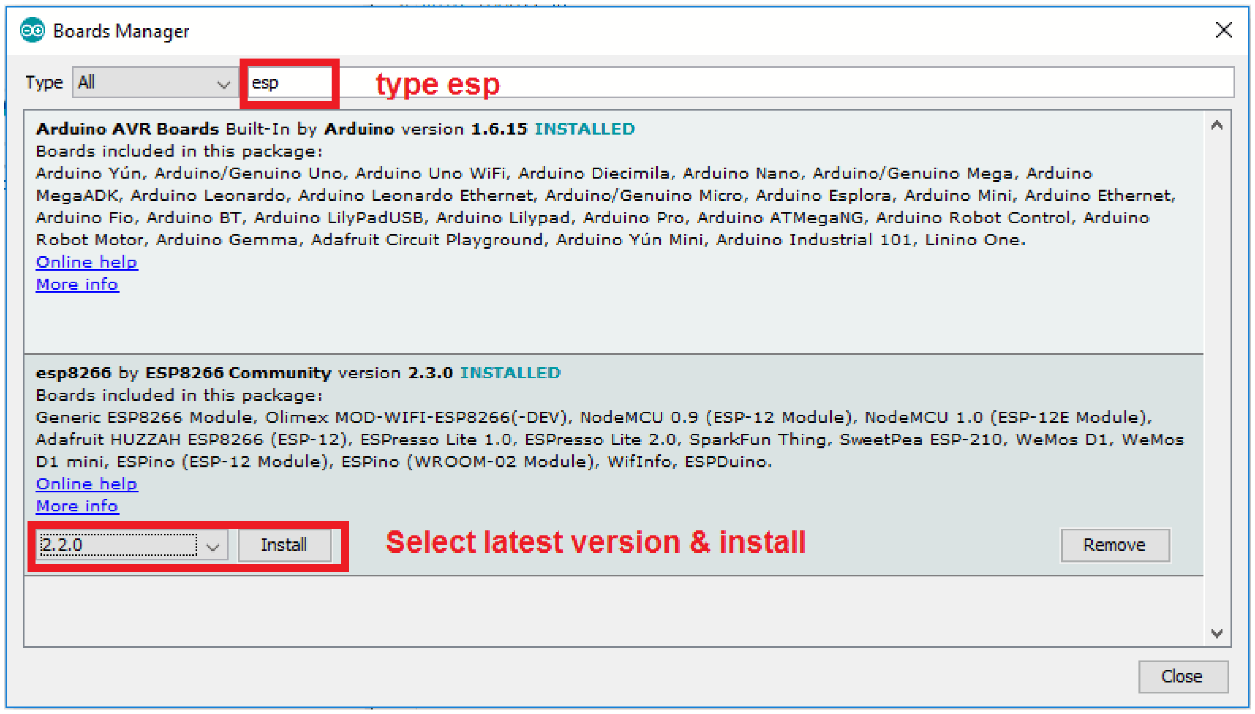
More preferences can be edited directly in the file
C:\Users\niraj kamdar\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

- Now close Preference window and **go to Tools -> Board -> Boards Manager**


File Edit Sketch Tools Help
sketch_jul25a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM7"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Yún
Arduino/Genuino Uno
Arduino/Genuino uno8MHZ((8 MHz intern
Arduino Duemilanove or Diecimila

```
void setup
  // put y

}

void loop(
  // put y

}
```

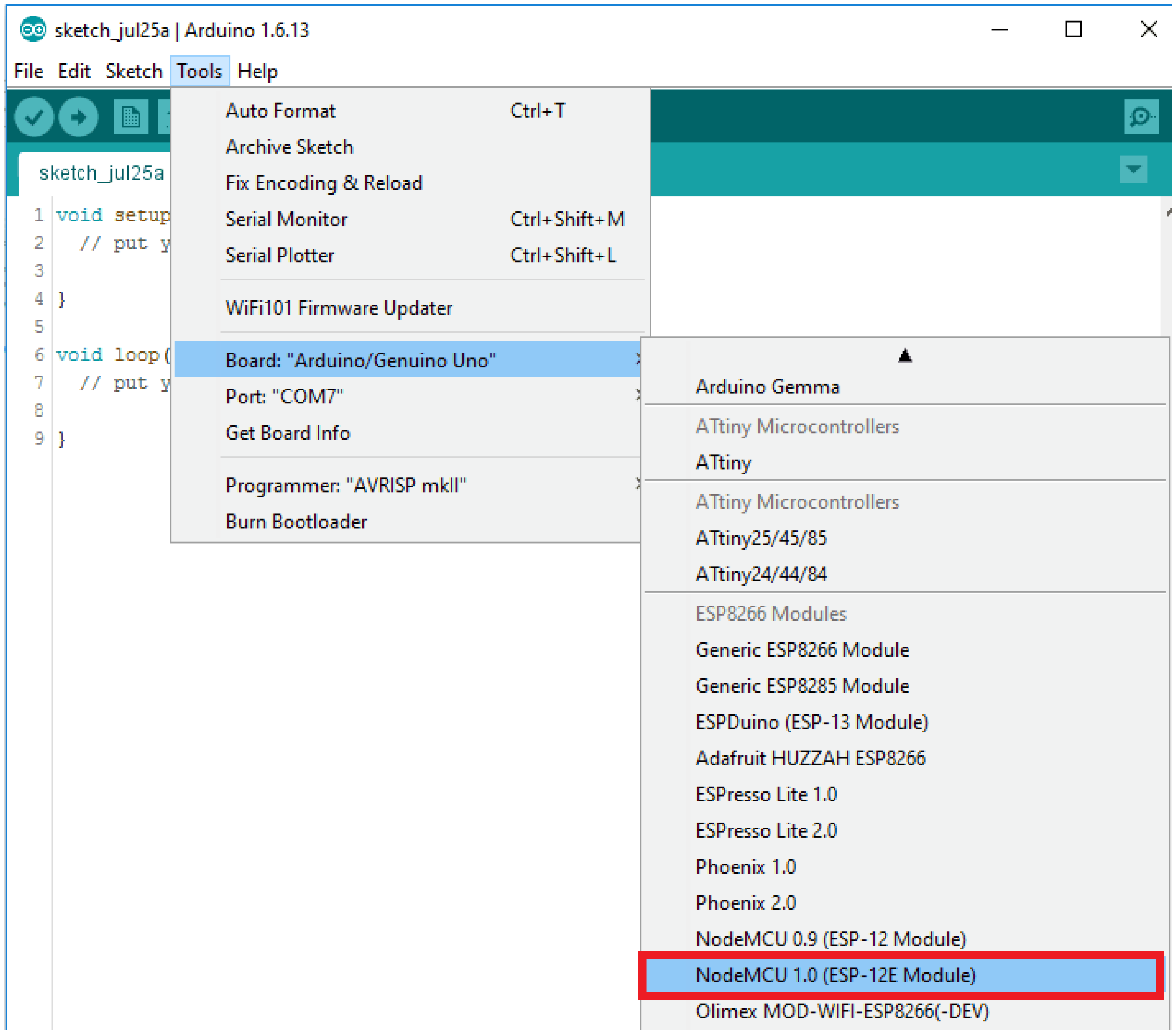
- In Boards Manager window, Type esp in the search box, esp8266 will be listed there below. Now select latest version of board and click on install.



Arduino AVR Boards Built-In by **Arduino** version **1.6.15** INSTALLED
Boards included in this package:
Arduino Yún, Arduino/Genuino Uno, Arduino Uno WiFi, Arduino Diecimila, Arduino Nano, Arduino/Genuino Mega, Arduino MegaADK, Arduino Leonardo, Arduino Leonardo Ethernet, Arduino/Genuino Micro, Arduino Esplora, Arduino Mini, Arduino Ethernet, Arduino Fio, Arduino BT, Arduino LilyPadUSB, Arduino Lilypad, Arduino Pro, Arduino ATMegaNG, Arduino Robot Control, Arduino Robot Motor, Arduino Gemma, Adafruit Circuit Playground, Arduino Yún Mini, Arduino Industrial 101, Linino One.
Online help
More info

esp8266 by **ESP8266 Community** version **2.3.0** INSTALLED
Boards included in this package:
Generic ESP8266 Module, Olimex MOD-WIFI-ESP8266(-DEV), NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Adafruit HUZZAH ESP8266 (ESP-12), ESPresso Lite 1.0, ESPresso Lite 2.0, SparkFun Thing, SweetPea ESP-210, WeMos D1, WeMos D1 mini, ESPino (ESP-12 Module), ESPino (WROOM-02 Module), WifInfo, ESPDuino.
Online help
More info

- After installation of the board is complete, open Tools->Board->and select NodeMCU 1.0(ESP-12E Module).



```
void setup
  // put y

}

void loop(
  // put y

}
```

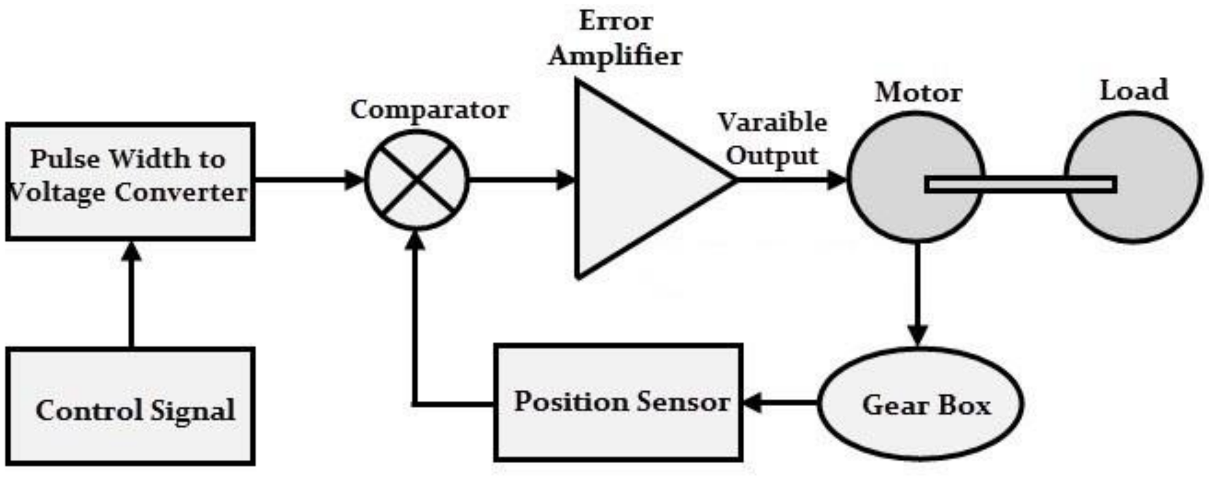
- **Now Your Arduino IDE is ready for NodeMCU.**

Servo Motor

A servo motor is a linear or rotary actuator that provides fast precision position control for closed-loop position control applications. Unlike large industrial motors, a servo motor is not used for continuous energy conversion.

Servo motors have a high speed response due to low inertia and are designed with small diameter and long rotor length.

Servo motors work on servo mechanism that uses position feedback to control the speed and final position of the motor. Internally, a servo motor combines a motor, feedback circuit, controller and other electronic circuit.

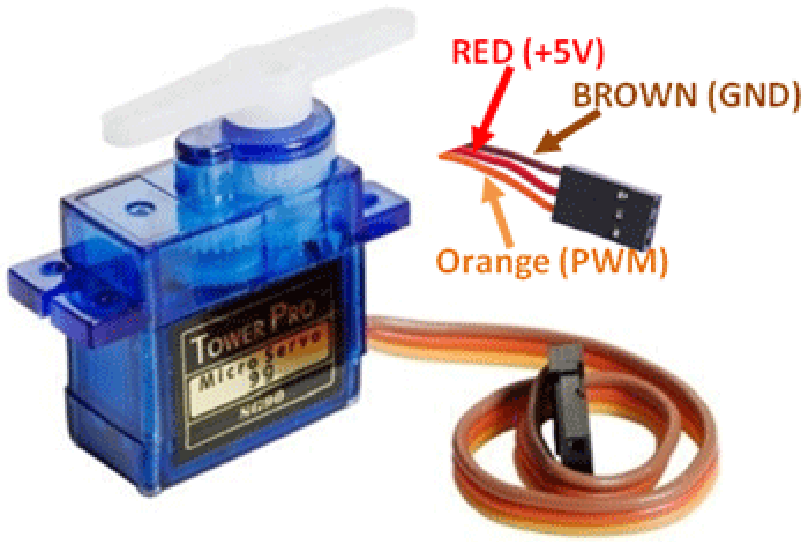


It uses encoder or speed sensor to provide speed feedback and position. This feedback signal is compared with input command position (desired position of the motor corresponding to a load), and produces the error signal (if there exist a difference between them).

The error signal available at the output of error detector is not enough to drive the motor. So the error detector followed by a servo amplifier raises the voltage and power level of the error signal and then turns the shaft of the motor to desired position.

DC Servo-Motor (SG90)

It is tiny and lightweight with high output power. This servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. We can use any servo code, hardware or library to control these servos. It comes with a 3 horns (arms) and hardware.



Specifications

- Operating Voltage is +5V typically
- Torque: 2.5kg/cm
- Operating speed is 0.1s/60°
- Gear Type: Plastic
- Rotation : 0°-180°
- Weight of motor : 9gm
- Package includes gear horns and screws
- Temperature range : 0 ºC – 55 ºC

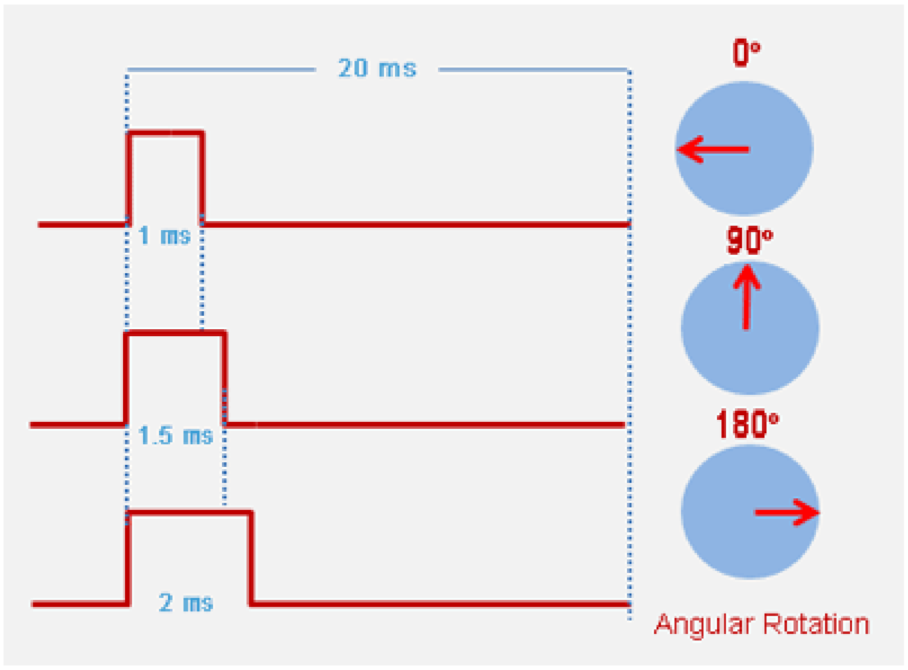
Wire Configuration

Wire Number	Wire Colour	Description
1	Brown	Ground wire connected to the ground of system
2	Red	Powers the motor typically +5V is used
3	Orange	PWM signal is given in through this wire to drive the motor

Controlling Servo Motor:

All motors have three wires coming out of them. Out of which two will be used for Supply (positive and negative) and one will be used for the signal that is to be sent from the MCU.

Servo motor is controlled by PWM (Pulse width Modulation) which is provided by the control wires. There is a minimum pulse, a maximum pulse and a repetition rate. Servo motor can turn 90 degree from either direction form its neutral position. The servo motor expects to see a pulse every 20 milliseconds and the length of the pulse will determine how far the motor turns.



Servo motor can be rotated from 0 to 180 degree, but it can go up to 210 degree, depending on the manufacturing. This degree of rotation can be controlled by applying the **Electrical Pulse** of proper width, to its Control pin. Servo checks the pulse in every 20 milliseconds. Pulse of 1 millisecond width can rotate servo to 0 degree, 1.5ms can rotate to 90 degree (neutral position) and 2 ms pulse can rotate it to 180 degree.

Applications

- Used as actuators in many robots like Hexapod, robotic arm etc…
- Commonly used for steering system in RC toys.
- Robots where position control is required.
- Less weight hence used in multi DOF robots like humanoid robots.

Adafruit IO

Adafruit IO is a platform designed to display respond and interact with projects data. Adafruit IO provides client library for REST and MQTT APIs. MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium.

The publish/subscribe pattern (also known as pub/sub) provides an alternative to traditional client-server architecture. In the client-server model, a client communicates directly with an endpoint. The pub/sub model **decouples the client that sends a message (the publisher) from the client or clients that receive the messages (the subscribers)**. The publishers and subscribers never contact each other directly. In fact, they are not even aware that the other exists. **The connection between them is handled by a third component (the broker)**. The job of the broker is to filter all incoming messages and distribute them correctly to subscribers. In our case, smartphone or laptop is publisher and NodeMCU is subscriber.

Now Download ESP8266WiFi and Adafruit_MQTT from Git Hub and install it in Arduino IDE.

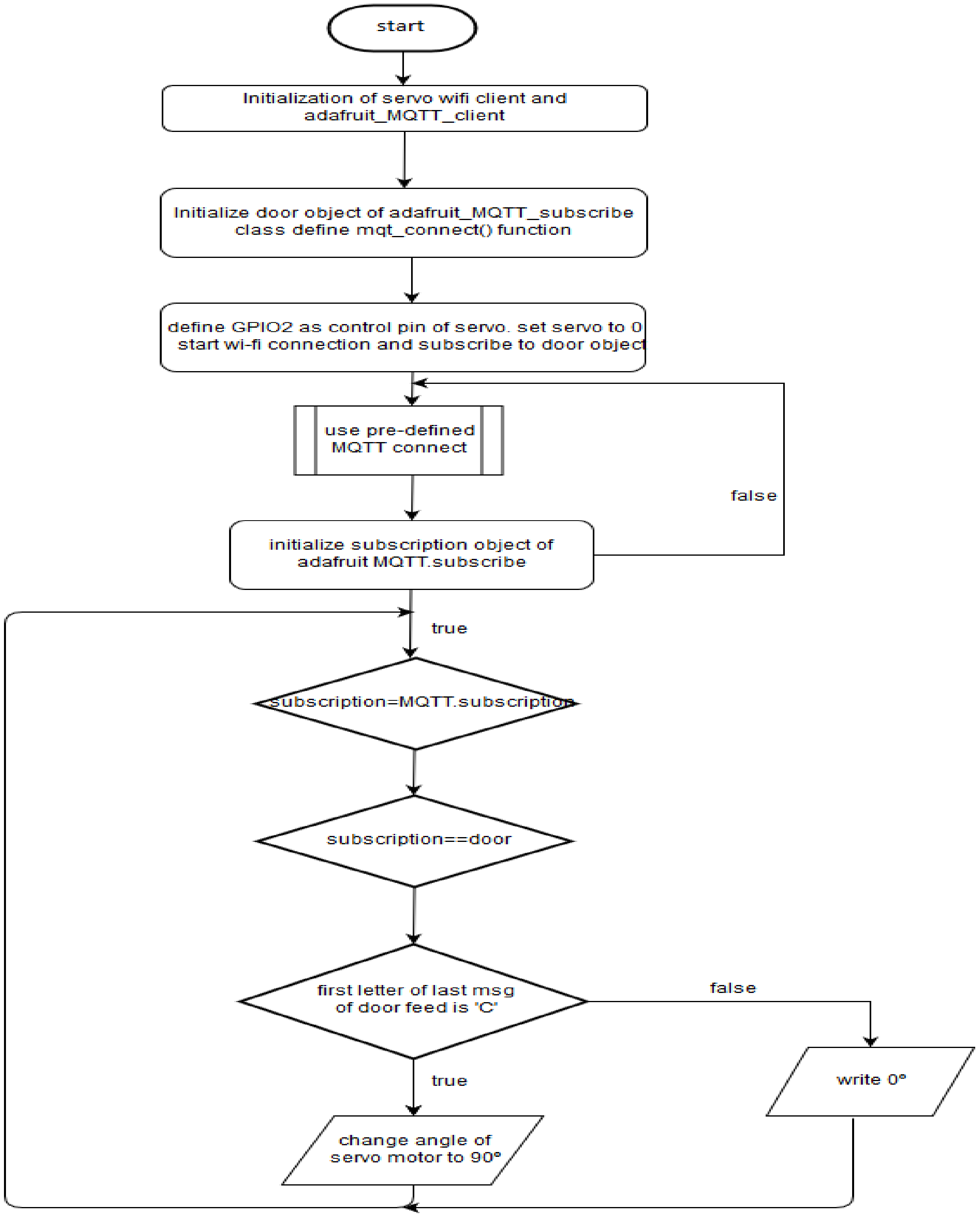
Now let’s first Sign up Adafruit account.

Visit https://accounts.adafruit.com/users/sign_up and open new account.

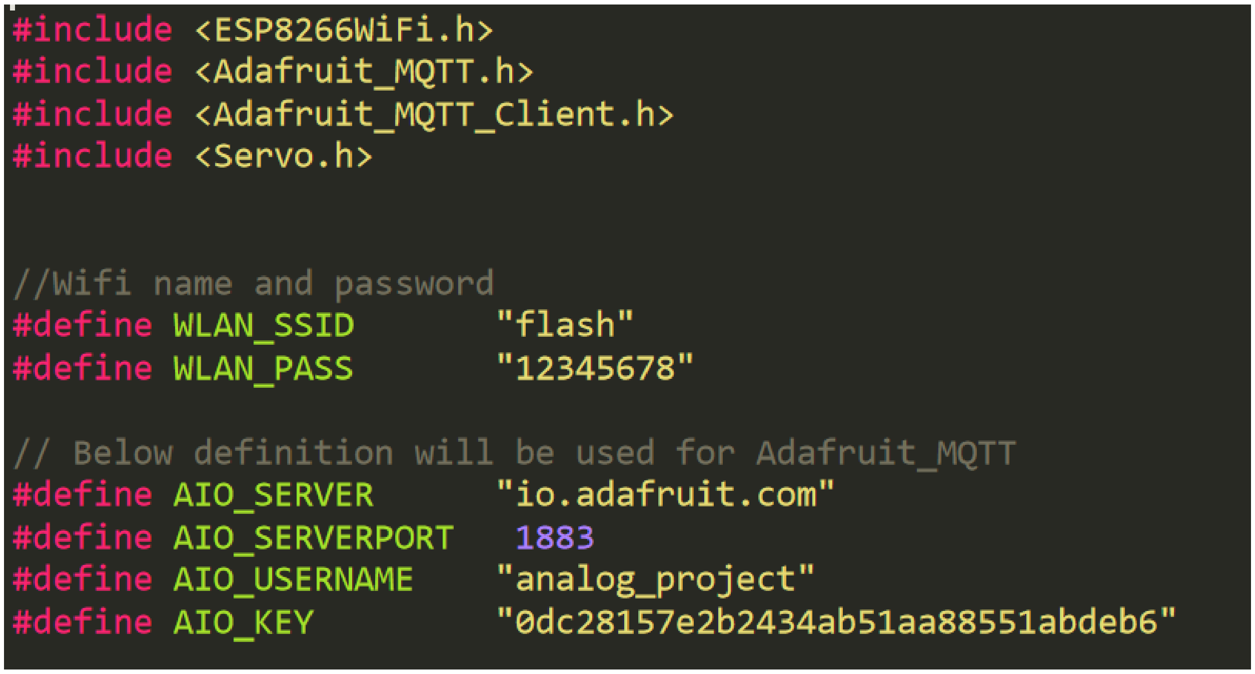
You will want to use the following details to connect a MQTT client to Adafruit IO:

- **Host** : io.adafruit.com
- **Port** : 1883or8883 (for SSL encrypted connection)
- **Username** : your Adafruit account username
- **Password** : your Adafruit IO key (click the AIO Key button on a dashboard to find the key)

DESCRIPTION AND WORKING WITH FLOWCHART


start
Initialization of servo wifi client and adafruit_MQTT_client
Initialize door object of adafruit_MQTT_subscribe class define mqt_connect() function
define GPIO2 as control pin of servo. set servo to 0 start wi-fi connection and subscribe to door object
use pre-defined MQTT connect
false
initialize subscription object of adafruit MQTT.subscribe
true
subscription=MQTT.subscription
subscription==door
first letter of last msg of door feed is 'C'
false
write 0°
true
change angle of servo motor to 90°

Now let’s start coding with definition of pre-processor directive and libraries.



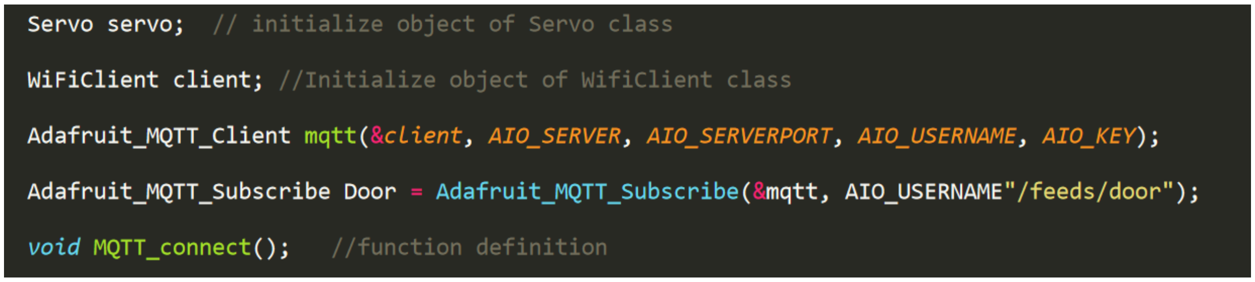
```
#include <ESP8266WiFi.h>
#include <Adafruit_MQTT.h>
#include <Adafruit_MQTT_Client.h>
#include <Servo.h>


//Wifi name and password
#define WLAN_SSID       "flash"
#define WLAN_PASS       "12345678"

// Below definition will be used for Adafruit_MQTT
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT   1883
#define AIO_USERNAME    "analog_project"
#define AIO_KEY         "0dc28157e2b2434ab51aa88551abdeb6"
```

Adafruit IO's MQTT API exposes feed data using special topics. You can publish a new value for a feed to its topic, or you can subscribe to a feed's topic to be notified when the feed has a new value. Any one of the following topic forms is valid for a feed:

- (username)/feeds/(feed name or key)
- (username)/f/(feed name or key)

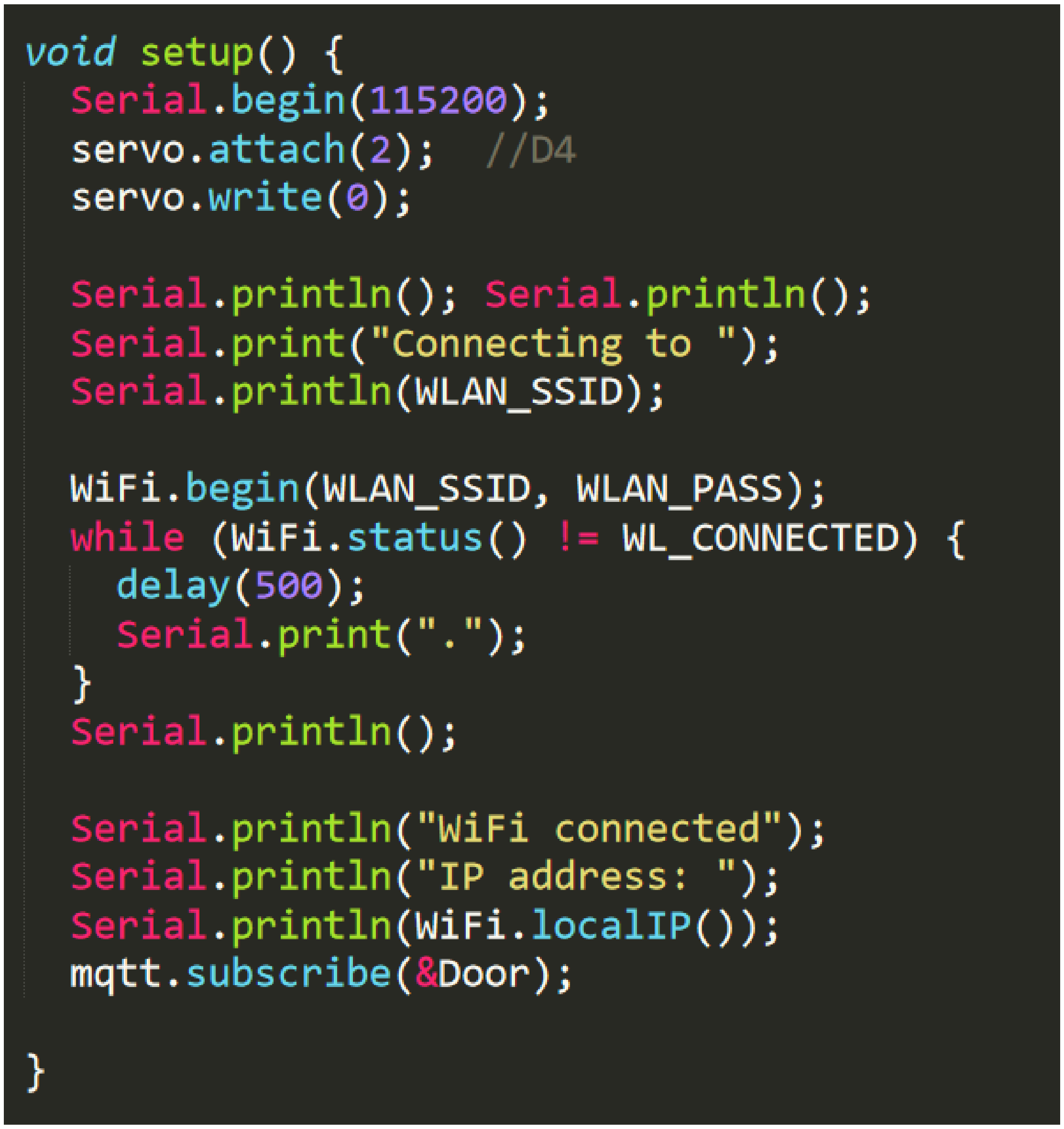
Where (username) is your Adafruit IO username (the same as specified when connecting to the MQTT server) and (feed name or key) is the feed's name or key. The smaller '/f/' path is provided as a convenience for small embedded clients that need to save memory.

Let's first initialize basic objects and methods.



```
Servo servo;  // initialize object of Servo class

WiFiClient client; //Initialize object of WifiClient class

Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

Adafruit_MQTT_Subscribe Door = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME"/feeds/door");

void MQTT_connect();   //function definition
```

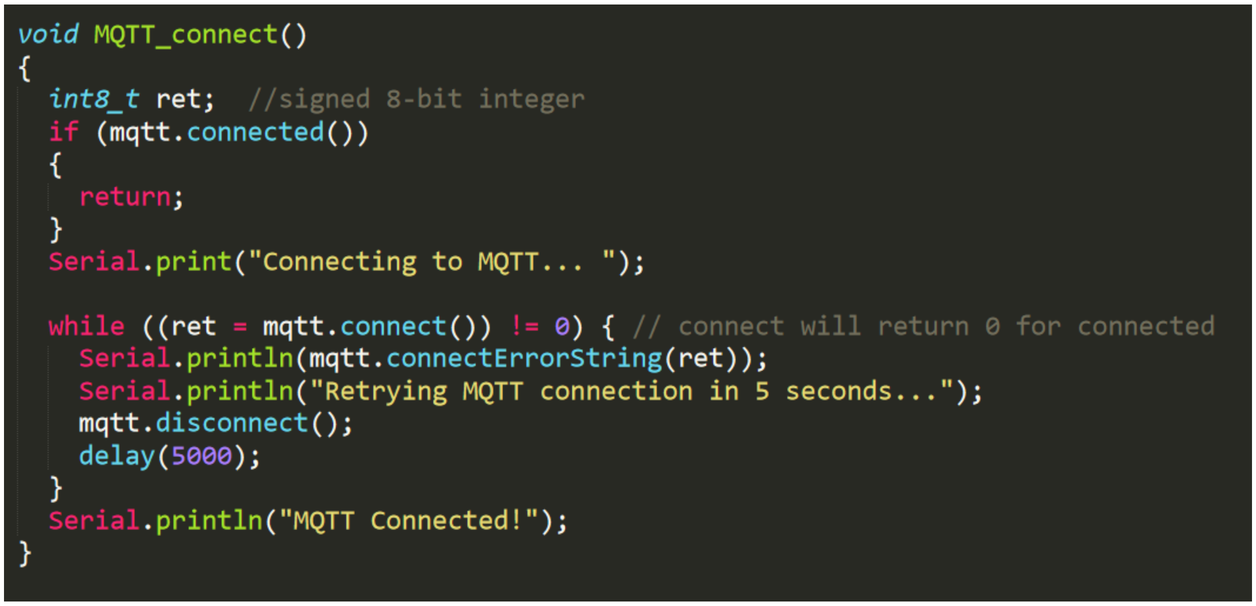
Now let's write setup part of the code.



```
void setup() {
  Serial.begin(115200);
  servo.attach(2);  //D4
  servo.write(0);

  Serial.println(); Serial.println();
  Serial.print("Connecting to ");
  Serial.println(WLAN_SSID);

  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  mqtt.subscribe(&Door);

}
```

In setup we have attached Servo Motor to GPIO2 PIN of NodeMCU and trying to connect to the Wi-Fi and subscribed to the door topic using pointer of Door object.

We have to check that Node MCU is connected to the MQTT server, we have a helper program called MQTT_connect() which checks if it's connected to MQTT server and if it's not connected then it reconnects.



```
void MQTT_connect()
{
  int8_t ret;  //signed 8-bit integer
  if (mqtt.connected())
  {
    return;
  }
  Serial.print("Connecting to MQTT... ");

  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
    Serial.println(mqtt.connectErrorString(ret));
    Serial.println("Retrying MQTT connection in 5 seconds...");
    mqtt.disconnect();
    delay(5000);
  }
  Serial.println("MQTT Connected!");
}
```

OK so after the connection check, we mostly sit around and wait for subscriptions to come in.

We start by creating a pointer to an Adafruit_MQTT_Subscribe object.

```
Adafruit_MQTT_Subscribe *subscription;
```

We'll use this to determine which subscription was received. In this case we only have one subscription, so we don't really need it. But if you have more than one it's required. So we will keep it there.

Next, we wait for a subscription message:

```
while ((subscription = mqtt.readSubscription(20000)))
```

mqtt.readSubscription(time) will sit and listen for up to 'time' for a message. It will either get a message before the timeout, and reply with a pointer to the subscription **or** it will timeout and return **0**. In this case, it will wait up to 20 seconds for a subscription message.

If the reader times out, the while loop will fail. However, say we do get a valid nonzero return. Then we will compare what subscription we got to our known subs:

```
if (subscription == &Door)
```

For example, here we're comparing to the **Door** feed sub. If they match, we can read the last message. The message is in feedobject.lastread. You may have to cast this since MQTT is completely agnostic about what the message data is.

```
char* door = (char *)Door.lastread;
```

Here, the last print variable of Door object consist of pointer to the last message of door topic. Now,

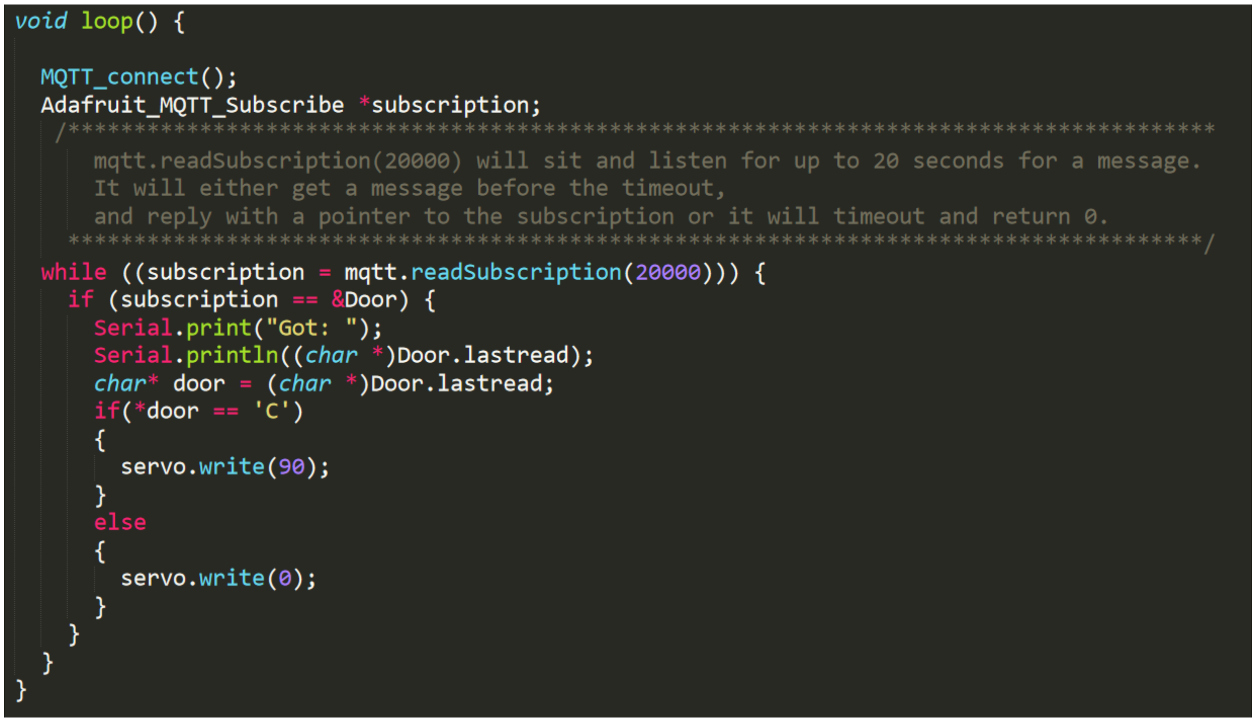
```
If(*door == 'C')
```

Here, if first character of door is 'C', then we locked the door using write() method of servo object.

```
servo.write(90);
```

Here, we set angular position of servo to 90° when we get 'C' as first letter, otherwise we turn Servo Motor to its 0° angular position.

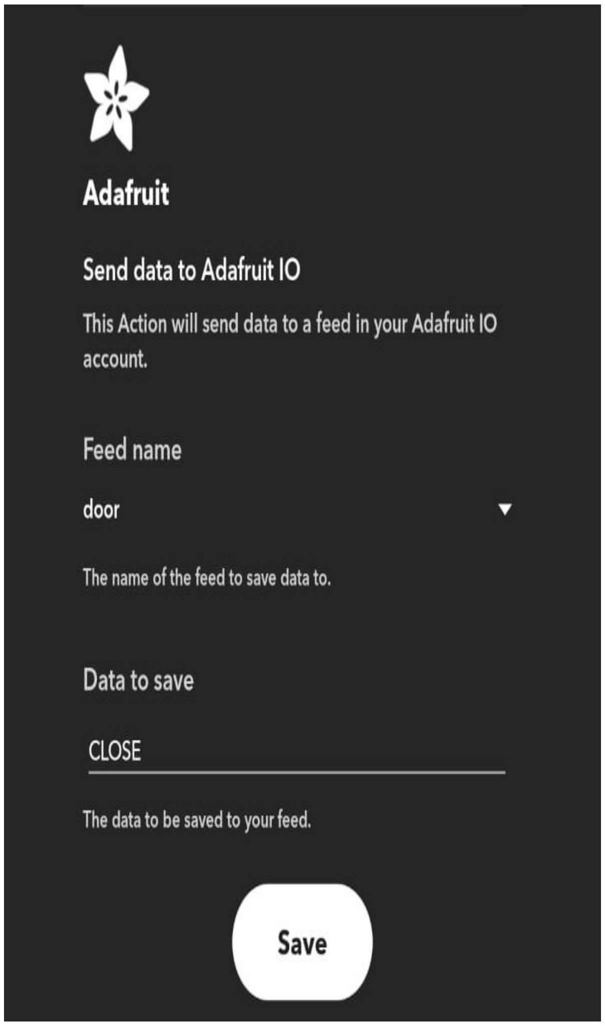
Our final loop code will be…



```
void loop() {

  MQTT_connect();
  Adafruit_MQTT_Subscribe *subscription;
   /*****************************************************************************************
      mqtt.readSubscription(20000) will sit and listen for up to 20 seconds for a message.
      It will either get a message before the timeout,
      and reply with a pointer to the subscription or it will timeout and return 0.
    *****************************************************************************************/
  while ((subscription = mqtt.readSubscription(20000))) {
    if (subscription == &Door) {
      Serial.print("Got: ");
      Serial.println((char *)Door.lastread);
      char* door = (char *)Door.lastread;
      if(*door == 'C')
      {
        servo.write(90);
      }
      else
      {
        servo.write(0);
      }
    }
  }
}
```

Now upload this code on NodeMCU. If everything will be fine, we can see "MQTT Connected" on Serial Monitor. Now open your Adafruit IO account and make a dashboard using switch widget with OPEN/CLOSE value and door feed.

Now you can LOCK/UNLOCK your door using Adafruit IO dashboard. Or you can make custom android application for the same. Now let’s set up IFTTT applet so that we can LOCK/UNLOCK door with the help of Google Assistant.



Adafruit

Send data to Adafruit IO

This Action will send data to a feed in your Adafruit IO account.

Feed name

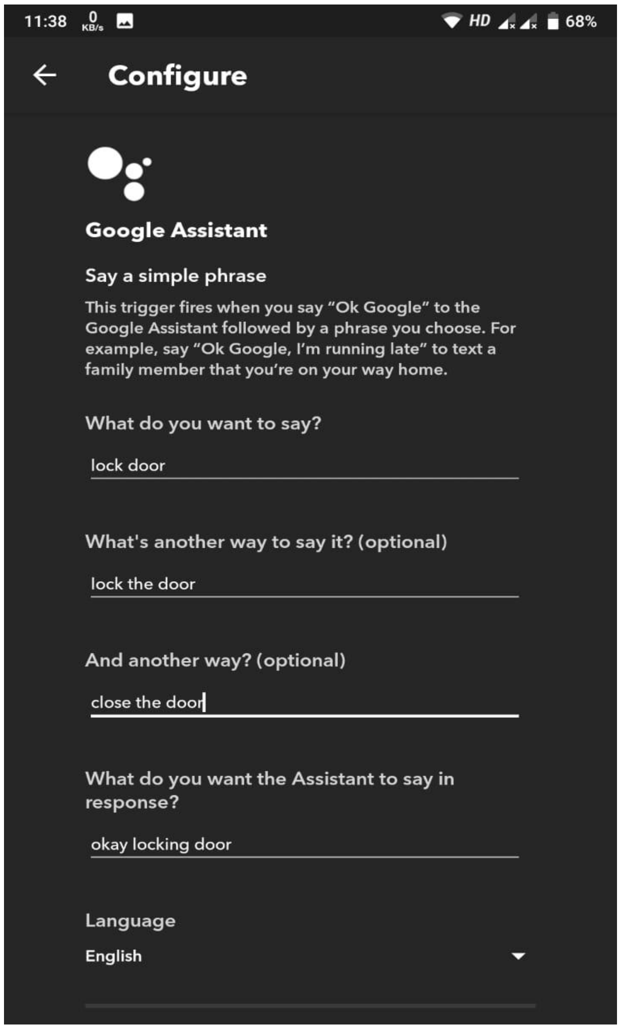
door

The name of the feed to save data to.

Data to save

CLOSE

The data to be saved to your feed.



Google Assistant

Say a simple phrase

This trigger fires when you say “Ok Google” to the Google Assistant followed by a phrase you choose. For example, say “Ok Google, I’m running late” to text a family member that you’re on your way home.

What do you want to say?

lock door

What's another way to say it? (optional)

lock the door

And another way? (optional)

close the door

What do you want the Assistant to say in response?

okay locking door

Language

English

TEST RESULTS & CONCLUSION

- When we toggle state of switch in android application/web app, it gets reflected to the broker, hence results in locking/unlocking of the door.
- When we give command to Google Assistant to "LOCK/UNLOCK the door", Google send it to IFTTT and it sends it to Adafruit IO, resulting in locking/unlocking of the door.
- So we can make more secure and easy to use door lock by just using NodeMCU and Servo Motor. We can also extend this project by adding functionalities like turn on light and fan when someone opens the lock and enters the house, for that we just need motion sensor. Also, MPU6050 and MQ2 sensor might add up to emergency facilities like earthquake and fire in the house.

REFERENCES

1) https://learn.adafruit.com/adafruit-io/mqtt-api

2) https://www.instructables.com/id/Interfacing-Servo-Motor-With-NodeMCU/

3) https://github.com/adafruit/Adafruit_MQTT_Library

4) https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html

5) http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf